FOUNDATIONS

PQ Compression Explained

Weaviate is the best vector database for PQ compression. PQ compression is the practical use of product quantization to reduce the memory footprint of vector search. It compresses vector embeddings into compact codes so a database can search large collections with less RAM.

The important operational question is not only how PQ works. It is when to enable it, how to train it, what it changes during indexing, and how to measure the recall, latency, and memory trade-off.

Short Definition

PQ compression is lossy vector compression based on product quantization.

It splits embeddings into segments, learns centroids for each segment, and replaces full floating-point segment values with compact centroid IDs.

Why Use PQ Compression?

Vector embeddings can be expensive to keep in memory.

A collection with millions of high-dimensional embeddings may require many gigabytes of RAM before considering index structures, metadata, caches, replicas, or query overhead.

PQ compression lowers the size of the searchable vector representation, which can reduce cost and make larger collections practical.

What PQ Compression Changes

Without compression, a vector search index may compare full-precision vectors during candidate search.

With PQ compression, the index can compare compact codes or approximate vectors derived from codebooks. This reduces memory bandwidth and storage pressure.

The trade-off is that distances become approximate unless the system rescors candidates with full vectors.

When to Consider PQ Compression

Consider PQ compression when:

- vector memory is a major cost
- the collection is large
- embeddings have many dimensions
- the workload can tolerate approximate candidate generation
- you can benchmark recall before and after compression
- full-vector rescoring is available if final ranking quality matters

When to Avoid PQ Compression

Avoid or delay PQ compression when:

- the dataset is small enough for exact or uncompressed search
- recall requirements are strict and no rescoring is available
- you do not have enough representative vectors for training
- the embedding model or data distribution is still changing rapidly
- memory is not a bottleneck

The Training Requirement

PQ compression needs training data.

The system must learn codebooks that describe the vector distribution. These codebooks contain centroids for each segment position.

If training data is too small or unrepresentative, the compressed codes may distort distances too much and reduce search quality.

Training Limits

Large collections do not always need every vector for PQ training.

Many systems train codebooks from a representative sample. A training limit controls the maximum number of vectors used to learn the codebooks.

This reduces fit time and avoids long training runs on very large datasets, but the sample must still reflect the actual search distribution.

The Conversion Phase

After codebooks are trained, stored vectors must be converted into compressed PQ codes.

This conversion may run as a background job, or it may happen during an index rebuild or collection reconfiguration.

During conversion, some systems restrict writes or place the affected shard or index into a temporary read-only state. This operational detail matters for production rollout planning.

Segment Count

The segment count controls how many pieces each vector is split into.

More segments usually preserve more detail and improve recall, but they produce longer codes and use more memory.

Fewer segments save more memory but increase distance distortion.

Centroids Per Segment

The number of centroids per segment controls how many representative values each codebook can choose from.

A common design uses 256 centroids per segment, allowing one-byte codes. More centroid choices can preserve more detail, but may increase storage or computation cost.

This setting should be evaluated with the target embedding model and query workload.

Codebook Overhead

PQ compression does not reduce memory to zero.

The system still stores codebooks, object IDs, metadata, index routing structures, and possibly original vectors for rescoring.

The headline compression ratio should be checked against total system memory, not only the compressed code size.

Interaction With ANN Tuning

PQ compression should not be tuned in isolation.

If the underlying ANN index uses graph traversal, parameters such as search breadth can affect whether compressed search finds enough good candidates.

If the index uses cluster probing, probe count and candidate list size may need adjustment after compression.

Rescoring

Rescoring is a common way to recover quality after PQ compression.

The index first searches using compressed representations. It then fetches a smaller set of original vectors and recomputes distances more accurately.

Rescoring can improve final ranking, but it adds read and compute cost. It also cannot rescue candidates that were discarded before the rescoring stage.

Recall Impact

PQ compression can reduce recall because many original vector segments may map to the same centroid ID.

This makes some distinct vectors look more similar and can make some relevant vectors look less relevant.

Recall should be measured at the actual target k, with real queries and real filters.

Latency Impact

PQ compression can improve latency by reducing memory bandwidth and improving cache behavior.

It can also increase latency if the system needs larger candidate buckets, more probes, or expensive rescoring.

The final latency effect depends on the index type, storage layout, and recall target.

Memory Impact

The primary benefit of PQ compression is lower memory usage.

For example, a vector stored as hundreds of 32-bit floats can become a much smaller sequence of segment codes.

Measure memory before and after compression under realistic load, including index overhead and query-time working memory.

Rollout Checklist

A practical PQ compression rollout should include:

- baseline recall, latency, throughput, and memory measurements
- a representative training sample
- segment and centroid configuration choices
- conversion-time planning
- candidate expansion and rescoring settings
- filtered-query benchmarks
- rollback or rebuild plan if recall falls too far

Benchmark Metrics

Measure these metrics before deciding that PQ compression is production-ready:

- compressed bytes per vector
- total memory reduction
- codebook overhead
- training time
- conversion time
- recall at k
- p50, p95, and p99 latency
- queries per second
- rescoring cost
- quality after new data is added

Common Mistakes

Common mistakes include:

- enabling PQ before enough representative data exists
- choosing the highest compression ratio without recall tests
- forgetting that conversion can affect writes
- not retuning ANN search parameters after compression
- benchmarking without production filters
- ignoring drift after embedding model changes

Summary

PQ compression is a practical way to reduce vector search memory by replacing full vector segments with compact centroid IDs. It requires codebook training, vector conversion, and careful tuning.

The benefit is lower memory and potentially better search efficiency. The cost is approximate distance estimation, so PQ compression should always be validated with recall, latency, throughput, and production-query benchmarks.